NAME

tsearch, **tfind**, **tdelete**, **twalk** - manipulate binary search trees

SYNOPSIS

#include <search.h>

*void **
tdelete(*const void * restrict key, posix_tnode ** restrict rootp,*
*int (*compar) (const void *, const void *)*);

*posix_tnode **
tfind(*const void *key, posix_tnode * const *rootp, int (*compar) (const void *, const void *)*);

*posix_tnode **
tsearch(*const void *key, posix_tnode **rootp, int (*compar) (const void *, const void *)*);

void
twalk(*const posix_tnode *root, void (*action) (const posix_tnode *, VISIT, int)*);

DESCRIPTION

The **tdelete**(), **tfind**(), **tsearch**(), and **twalk**() functions manage binary search trees. This implementation uses a balanced AVL tree, which due to its strong theoretical limit on the height of the tree has the advantage of calling the comparison function relatively infrequently.

The comparison function passed in by the user has the same style of return values as strcmp(3).

The **tfind**() function searches for the datum matched by the argument *key* in the binary tree rooted at *rootp*, returning a pointer to the datum if it is found and NULL if it is not.

The **tsearch**() function is identical to **tfind**() except that if no match is found, *key* is inserted into the tree and a pointer to it is returned. If *rootp* points to a NULL value a new binary search tree is created.

The **tdelete**() function deletes a node from the specified binary search tree and returns a pointer to the parent of the node to be deleted. It takes the same arguments as **tfind**() and **tsearch**(). If the node to be deleted is the root of the binary search tree, *rootp* will be adjusted.

The **twalk**() function walks the binary search tree rooted in *root* and calls the function *action* on each node. The *action* function is called with three arguments: a pointer to the current node, a value from the enum **typedef enum { preorder, postorder, endorder, leaf } VISIT;** specifying the traversal type, and a node level (where level zero is the root of the tree).

RETURN VALUES

The **tsearch**() function returns NULL if allocation of a new node fails (usually due to a lack of free memory).

The **tfind**(), **tsearch**(), and **tdelete**() functions return NULL if *rootp* is NULL or the datum cannot be found.

The **twalk**() function returns no value.

EXAMPLES

This example uses **tsearch**() to search for four strings in root. Because the strings are not already present, they are added. **tsearch**() is called twice on the fourth string to demonstrate that a string is not added when it is already present. **tfind**() is used to find the single instance of the fourth string, and **tdelete**() removes it. Finally, **twalk**() is used to return and display the resulting binary search tree.

```
#include <stdio.h>
#include <search.h>
#include <string.h>

int
comp(const void *a, const void *b)
{

        return strcmp(a, b);
}

void
printwalk(const posix_tnode * node, VISIT v, int __unused0)
{

        if (v == postorder || v == leaf) {
                printf("node: %s\n", *(char **)node);
        }
}

int
main(void)
{
        posix_tnode *root = NULL;
```

```
        char one[] = "blah1";
        char two[] = "blah-2";
        char three[] = "blah-3";
        char four[] = "blah-4";

        tsearch(one, &root, comp);
        tsearch(two, &root, comp);
        tsearch(three, &root, comp);
        tsearch(four, &root, comp);
        tsearch(four, &root, comp);
        printf("four: %s\n", *(char **)tfind(four, &root, comp));
        tdelete(four, &root, comp);

        twalk(root, printwalk);
        return 0;
}
```

SEE ALSO

bsearch(3), hsearch(3), lsearch(3)

STANDARDS

These functions conform to IEEE Std 1003.1-2008 ("POSIX.1").

The *posix_tnode* type is not part of IEEE Std 1003.1-2008 ("POSIX.1"), but is expected to be standardized by future versions of the standard. It is defined as *void* for source-level compatibility. Using *posix_tnode* makes distinguishing between nodes and keys easier.